# From multi-physics models to neural network for predictive control synthesis

Pierre Clément Blaud, Philippe Chevrel, Fabien Claveau, Pierrick Haurant, Anthony Mouraud

# From multi-physics models to neural network for predictive control synthesis

Pierre Clément Blaud[a,b], Philippe Chevrel[b], Fabien Claveau[b], Pierrick Haurant[c], Anthony Mouraud[a]

[a]*CEA, CEA Tech Pays de la Loire, F-44340 Bouguenais, France*
[b]*IMT Atlantique, LS2N, UMR CNRS 6004, F-44307 Nantes, France*
[c]*IMT Atlantique, GEPEA, UMR CNRS 6144, F-44307 Nantes, France*

**Abstract**

The aim of this document is to present an efficient and systematic method of model-based predictive control synthesis. Model predictive control requires using a model of a dynamical system, that can be linear, time-varying, non-linear or identified from data. Finding a model that is both precise and simulatable at low computational cost can be challenging and time consuming due to requiring extensive knowledge of the system and physics as well as a large volume of data with relevant scenarios and sometimes a complicated identification work (filtering noises and bias, data formatting, etc.). The proposed methodology begins with fine-scale multi-physics modelling, which is possible thanks to open model libraries (see Modelica). The obtained model is then simulated by considering ad hoc scenarios to generate data, which are then used to identify a neural network, that will support the predictive control syntheses. The systematic methodology is detailed and applied to the widely used control benchmark known as the quadruple tanks process. Results show that the methodology is accurately applied to optimize hyperparameters in finding a neural network model and to control the quadruple tanks process with the predictive controller.

*Keywords:* Feedforward neural network, hyperparameters optimisation, economic model predictive control, modelica, model identification.

## 1. Introduction

Systems and control technology are everywhere, from homes to airplanes, and they face great challenges in areas, such as water networks, increase share of renewable energy, intelligent transport, process engineering or medical science. Some challenges require interdisciplinary methods ranging from advanced control techniques, modelling interconnected complex systems and machine learning [40]. Among advanced control techniques, Model Predictive Control (MPC)

*Email addresses:* `pierre-clement.blaud@imt-atlantique.fr` (Pierre Clément Blaud), `philippe.chevrel@imt-atlantique.fr` (Philippe Chevrel)

is commonly studied in academia [49] and used in industrial applications [11], because it enables considering constraints on the state variables of the process. Since initial implementation in the petroleum industry [56], MPC has broadened its field of industrial applications to include traffic control [71], energy management in residential house [42], heating ventilation and air-conditioning control [3], aerospace [65] or the automotive industry [29]. Classical MPC involves a state space model of the dynamical system, states constraints, inputs constraints and a quadratic cost function to minimize. The optimization is performed online and the first optimal control sample is the one applied to the actuators, while a new optimization is relaunched to prepare the next step. There are some extensions or variants of MPC, such as Economic Model Predictive control (EMPC), where an economic cost is substituted with the quadratic one in order to control the dynamical system for economic behavior [55, 17].

Finding a suitable dynamical model allowing predictive control synthesis poses a challenging task for control engineering. Models that finely represent the process are generally complicated, which makes it difficult to use them for MPC, and model simplification is often required. To acquire a suitable model, one method is to linearize the physical model around an operative point, which allows controller tuning and leads to linear MPC [51] or linear EMPC [36]; however, the model poorly represents the non-linear dynamical system. To increase model accuracy, a linear time-varying model from the non-linear model can be adopted [50] or more directly the non-linear model [25]. Furthermore, a hybrid MPC is possible to handle systems with continuous states, logic rules, discrete states or if-then-else conditions [41]. The engineering time required for the modelling process of modelling is labour intensive and may be out of proportion based on industrial expectations [46].

One method of reducing engineering time is to identify the model from input-output signals of the process to achieve a black-box model. Black-box models used for predictive controllers can be canonically parametrized model or a neural network. An AutoRegressive Exogenous (ARX) model is presented in [30], while a recurrent neural network is used by an MPC in [67] and also in [68] with an EMPC. In [60], the authors compared parametric models and neural networks models to identify a cooling system for a business center and found that the neural network model is more accurate compared to a parametric one. In [67], the data were generated from the non-linear state space model, by simulation. In [60], the data originate from experimental measures and yield favorable results; however, there are many cases where measured data are lacking or expensive [57].

The method considered in this work is intended for dynamical systems to be controlled, for which measured data are lacking, and therefore data will be generated via simulation using multi-physical modeling. Model identification is then performed to acquire a model which requires less calculation time for its simulation while maintaining accuracy. The following method includes several advantages: First, when using a simulation model, a dedicated researcher or engineer performs physical modelling using specific simulation tools, which makes the simulation model accurate regarding the state of the art. Second, this method reduces engineering time for the control researchers in finding a black-box model rather than a classical physical model for a predictive controller. Third, this method helps finding a black-box model with greater accuracy than classical physical models. Fourth, in some real systems, there are insufficient measurement data to identify an accurate black-box model, this method helps in generating data. Moreover, many simulation tools represent dynamical system in detail, such as the Modelica language [48], which allows modeling multi-physical networked systems such as mechanical, electrical, thermal, hydraulic, pneumatic and fluid. In addition, commercial or open-source libraries model many systems, such as electrical [20], buildings [66], power plant system [27], district heating system [22] or greenhouses [5].

Despite the popularity of i) multi-physics modeling for simulation on the one hand, ii) neural network modeling and iii) predictive control method, the current literature seemed to the authors not to make the most of the recent advances on these subjects, in particular by not proposing an integrated methodology for the global synthesis of control laws. The main contribution of this work comes in the form of a methodology systematically addressing the issue of predictive controller synthesis relying on a dynamic model based on a neural network for its implementation, this model being identified from data from multi-physical simulations (in the absence of big measurement data). The identification of the process proceeds from hyperparameters optimized with a meta-heuristic algorithm to obtain the ad hoc neural network. The predictive controller thus deduced from the application of the proposed methodology was successfully applied to a conventional control benchmark. In summary, the methodology proposed in this article links:

- The use of multi-physics modeling by taking advantage of its ability to combine physical subsystems of different types as well as the capitalization and reuse of such models thanks to shared libraries;
- The possibilities of identifying neural networks based non-linear dynamic models, with an automated choice of hyperparameters, the latter being heuristically optimized according to explicit criteria;
- The possibilities offered in control theory by non-linear model-based predictive control;

The paper is organized as follows: in Section 2, main elements of the methodology are depicted; in Section 3, the methodology is detailed; Section 4 presents the application of the methodology to a classical control benchmark; the results are depicted in Section 5; Section 6 concludes the work.

## 2. Methodology highlights

This study's methodology is shown in Fig. 1 and divided into five stages. This section provides a tutorial to aid understanding and implementation.

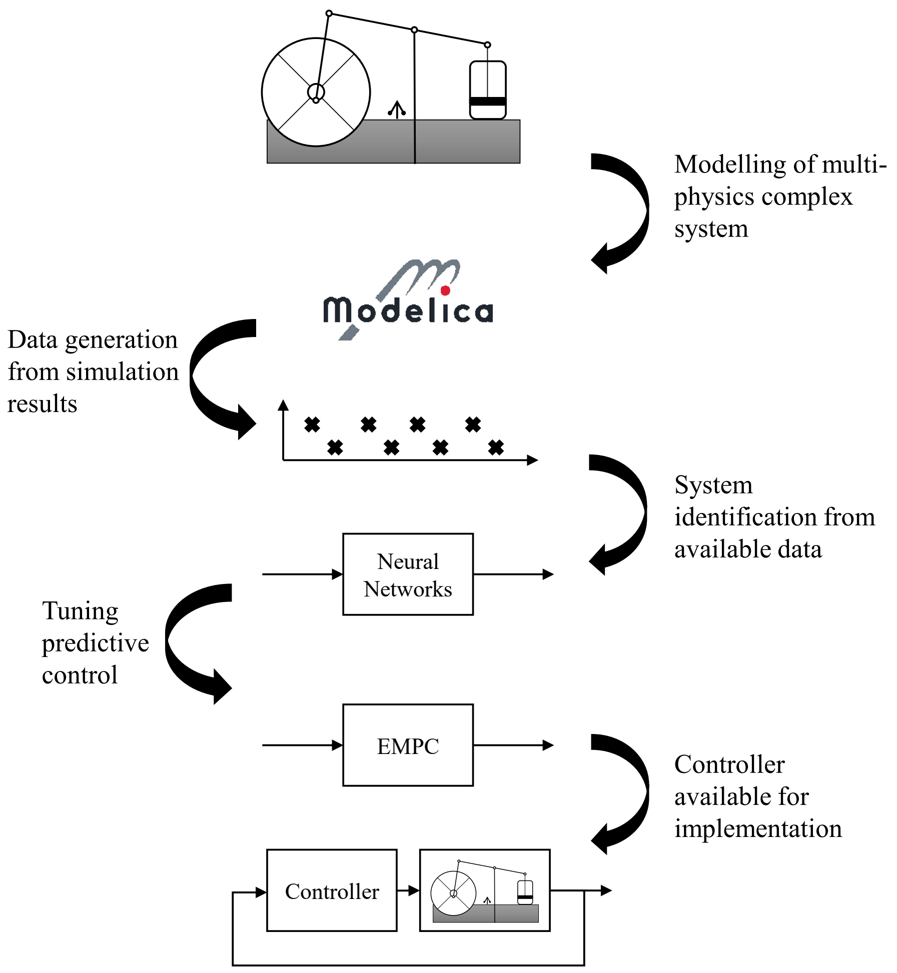
Modelling of multi-physics complex system
Modelica
Data generation from simulation results
System identification from available data
Neural Networks
Tuning predictive control
EMPC
Controller available for implementation
Controller

Figure 1: Schematic view of the methodology.

The first step regards the fine-scale modelling using multi-physical modelling tools, which can be performed by modelling experts to achieve a model with the closest resemblance. In addition, the model's parameters can be scaled using known completed plant parameters and using calibration from plant data measures if some are missing. Information about complex networked physical systems modeling is provided in Section 3.1.

The second step involves data generated from the simulation model designed in the first step. The model is extensively simulated from selected scenarios to best cover the conditions of the system's use to be controlled, and the data produced must be both qualitative and quantitative to obtain a representative neural network (by optimization). These aspects are discussed in Section 3.2.

The third step concerns identifying the neural network approximating the

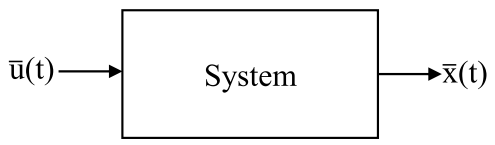

Figure 2: System input-output representation with inputs $\bar{u}(t)$ and outputs $\bar{x}(t)$.

dynamic behavior of the process (see Section 3.3). The accuracy of the targeted predictive control depends on the quality of this model, which will be used as an internal model. The description of this model and methods analyzing its stability will be specified, and an automated tuning method of an appropriate set of hyperparameters will be introduced to aid for building and optimizing the neural network.

The fourth step regards with establishing the predictive control. In this work, the predictive control considered is economic, although the methodology could be applied to a classical MPC with a quadratic cost function. Section 3.4 introduces the EMPC (cost and constraint function) and specifies using the neural model from step three. In addition, some computer tools for implementing the controller and resolving the optimization problem will be detailed.

All steps are detailed in the following sections.

## 3. Methodology

### *3.1. Modelling complex systems*

The modelling of complex networked physical systems has been a focus for years through formalism as bond graph [21] or Energetic Macroscopic Representation (EMR) [4] that enable graphical representation. Both are based on energy flows, effort and power relation, and physical relations use time-dependent Ordinary Differential Equation (ODE) supporting 0-D modelling. In addition, a computing language such as Modelica allows modeling and simulation based on a uniform general language for model design. A graphical representation of each sub-system and their interactions may be used, either from bond graph [62] or Modelica's graphical representations. Each block and subsystem are assembled (graphically or not) to build the whole system [48]. A Modelica code concatenating all the physical equations is produced, which enables numerically simulation using a numerical solver such as a Differential Algebraic System Solver (DASSL) [45].

### *3.2. Data generation*

Generating simulation data is mandatory for system identification. Before generating data, the control actuators must first be selected and will form the inputs of the identified dynamical system; second, the measures must be selected and will form the states of the identified dynamical system. As a result, we will use an input-output description of the system (see Fig. 2). Both actuators and measures must be chosen wisely according to the dynamical system capabilities and with physical implementation possibilities. After inputs and outputs are chosen, the signals sequences are applied to the inputs while the outputs are measured. The characteristics of input sequences of the simulated dynamical system require careful selection to generate output data that reveal its dynamic nature, which is called persistent excitation [57]. Various inputs could be used to excite the system, such as constant input signals (on-off), ramps, steps, sinusoidal, random sequences or Pseudo-Random Binary Sequences (PRBS) [57, 39]. These sequences lack the same persistent excitation. In [57], the authors found that random sequences adequately excite the system; in [39], the authors indicate that PRBS can excite the dynamical system with distinct sinusoidal components. In addition, a sequence that is a mix of the random sequence and PRBS is called a Multilevel Pseudo-Random Signal (MPRS).

### *3.3. Identification*

#### *3.3.1. Dynamical system*

In this work, we considered a discrete time invariant system of the following form:

$$\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k) \tag{1}$$

where $\bar{x} \in \bar{\mathbf{X}}^{n_{\bar{x}}}$ is the system state, $\bar{u} \in \bar{\mathbf{U}}^{n_{\bar{u}}}$ is the command system input and $k \in \mathbf{I}_{\geq 0}$ is the discrete sampling time occurrences. Moreover, all states are available for the controller.

#### *3.3.2. Feedforward neural network*

The first mathematical model of a neuron was proposed by W. McCulloch and W. Pitts in 1943 and is called the McCulloch-Pitts model [70]. The mathematical formulation of single artificial neuron with multi-inputs is equal to [70]:

$$a = \sigma(Wp + b) \tag{2}$$

where $a$ is the output of the neuron, $\sigma$ is the activation function, $W$ is the weight matrix, $p$ is the input of the neuron and $b$ is the bias. One neuron is clearly insufficient to extract interesting characteristics from the data, which are usually stacked and leads to a Feedforward Neural Network (FNN) [37]. A schematic representation of FNN is shown in Fig. 3. FNNs represent the simplest neural network architectures and were chosen for implementation in this work. The mathematical formulation of a FNN with $i$ hidden layers with the purpose to identify the dynamical system from Eq. (1) is equal to:

$$x_{k+1} = f_{h_1}(v_k) \circ f_{h_2}(a_1) \circ \cdots \circ f_{h_{i-1}}(a_{i-2}) \circ f_{h_i}(a_{i-1}) \tag{3}$$

where $x_{k+1}$ is the predicted output of the dynamical system by the neural network, and $f_{h_i}$ is the hidden layer, such as $a_i = f_{h_i}(p_i) = \sigma^i(W^i p^i + b^i)$ with $a_i$ the output and $p_i$ the input. In addition, $\sigma^i$ is the activation function, $W^i$ is the weighting matrix and $b^i$ is the bias. Then, $v_k$ is the input of the FNN, which is the concatenation of the dynamical system inputs $\bar{u}_k$ and states $\bar{x}_k$ at current and past points in time, $v_k = \text{cat}(\bar{x}_k, \bar{u}_k, \bar{x}_{k-1}, \bar{u}_{k-1}, ..., \bar{x}_{k-l}, \bar{u}_{k-l})$, with $l$ the back horizon as number of lag.

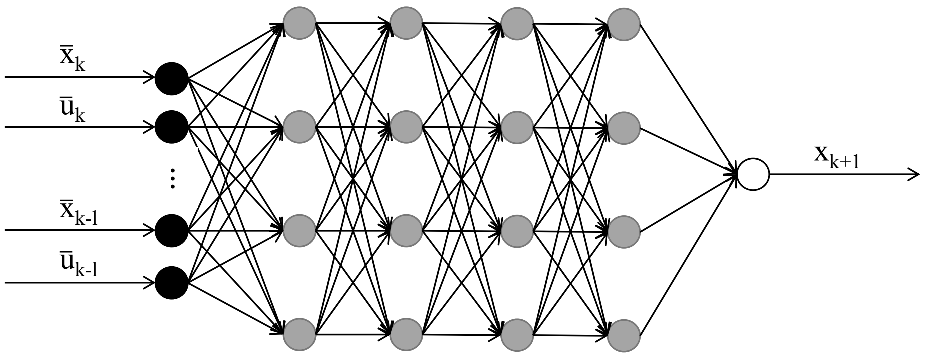

Figure 3: Schematic view of a feedforward neural network with four hidden layers, inputs layer neurons are black, hidden layers neurons are grey, and output layer is white.

*3.3.3. Fidelity measure*

The loss function is employed to measure the quality of the FNN to reproduce a signal. This provides a quantitative scoring that shows the degree of similarity between from the data and the FNN output [64]. Several loss functions are available, such as the Mean Square Error (MSE):

$$f_{loss} = \frac{1}{Nb}\sum_{i=1}^{Nb}[x_i - \bar{x}_i]^2 \tag{4}$$

where $Nb$ is the number of samples, $x_i$ is the neural network output and $\bar{x}_i$ is the target. Other loss functions include Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) [43].

*3.3.4. Hyperparameters tuning*

The hyperparameters defining the neural network influence its performance and include the number of hidden layers, number of neurons per hidden layers, type of activation function, batch size, epochs, the optimizer used and its own parameters [69]. For widelyd use accelerated gradient descent optimizers, parameters include the learning rate and momentums [72]. It is difficult to tune hyperparameters since they offer numerous choices, and we propose to automate these choices by selecting them via optimization, therefore at a higher hierarchical level, to minimize the final loss function. Performing optimization is challenging since it is non-differentiable and involves mixed and constrained variables. The optimization problem is formalized in the form:

$$
\begin{aligned}
\min_{\chi} \quad & f_{loss}(\chi) && \text{(5a)}\\
\text{subject to} \quad & f_{nn}(\sigma, \delta, \theta, \mu, \nu, o, \tau, \upsilon, \phi) && \text{(5b)}\\
& \sigma \in \Sigma && \text{(5c)}\\
& \delta \in \Delta && \text{(5d)}\\
& \theta \in \Theta && \text{(5e)}\\
& \mu \in M && \text{(5f)}\\
& \nu \in N && \text{(5g)}\\
& o \in O && \text{(5h)}\\
& \tau \in T && \text{(5i)}\\
& \upsilon \in \Upsilon && \text{(5j)}\\
& \phi \in \Phi && \text{(5k)}
\end{aligned}
$$

where $f_{loss}$ is the loss function for a given data set. $f_{nn}$ is the neural network architecture with dedicated inputs and outputs and Eq. (5b) is the related constraint. $\sigma$ is the activation function and $\Sigma$ is the constraint, see Eq. (5c). $\delta$ is the number of neurons per hidden layer and $\Delta$ is the related constraint, see Eq. (5d). $\theta$ is the number of hidden layer and $\Theta$ is the hidden layer constraint, see Eq. (5e). $\mu$ is the epoch number and $M$ is the constraint, see Eq. (5f). $\nu$ is the batch size and $N$ is the constraint, see Eq. (5g). $o$ is the optimizer and $O$ is the constraint, see Eq. (5h). $\tau$ is the optimizer learning rate and $T$ is the related constraint, see Eq. (5i). $\upsilon$ is the optimizer momentum exponential decay and $\Upsilon$ is the constraint, see Eq. (5j). $\phi$ is the optimizer momentum estimate and $\Phi$ is the related constraint, see Eq. (5k). Finally $\chi$ is the vector of optimization variables.

#### *3.3.5. Implementation on computer machines*

FNNs are implemented within machine learning tools such as PyTorch [52] or Flux [32], which help reduce implementation complexity and allow training neural networks with Graphical Processing Unit (GPU) to accelerate training. In addition, a meta-heuristic algorithm is used to solve the optimization problem Eq. (5) due to the problem's of non-differentiability. The era of meta-heuristics features abundant algorithms for finding the best set of hyper-parameters, such as genetic algorithms or particle swarm optimization [1]. In addition, meta-heuristic algorithms are available in language tools such as DEAP [19] or BlackBoxOptim.jl [18]. The implementation proposed in this paper is discussed in Section 4.2.

### *3.4. Economic Model Predictive Control*

#### *3.4.1. Description*

There are two types of EMPC: The first involves a terminal cost and/or terminal states constraints [17], while the second has no terminal cost and constraints [24] and is employed in this work. The cost function used is moreover not quadratic, but associated with a so-called “economic” cost (Economic MPC). Traditionally, approaches seeking to guarantee the stability of MPC controls, introduce a constraint or a terminal cost. Traditionally, approaches seeking to ensure the stability of MPC controls introduce a constraint or a terminal cost. Here, a contractive constraint was added on the terminal state to ensure the MPC to be stabilized [13]. The formulation is described below, but please note that neural network input $v_k$ is reworded with $x_k$, $u_k$ for notation simplicity :

$$\begin{aligned}
\min_{x,u} \quad & \sum_{k=0}^{N-1} l_e(x_k, u_k) && \text{(6a)}\\
\text{subject to} \quad & x_{k+1} = f^*_{nn}(x_k, u_k) && \text{(6b)}\\
& x_0 = \bar{x}(t) && \text{(6c)}\\
& x \in \mathcal{X} && \text{(6d)}\\
& u \in \mathcal{U} && \text{(6e)}\\
& \tilde{x}'_0 P \tilde{x}_0 \leq \tilde{x}'_N P \tilde{x}_N && \text{(6f)}
\end{aligned}$$

where Eq. (6a) is the economic cost to minimize and N is the number of samples defining the length of the considered time horizon. Eq. (6b) is the time evolution constraint of the state, which is based on the best neural network architecture found in Section 3.3.4. Let’s note that optimization problem (6) is non-linear, in particular, Eq. (6b) is non-linear due to FNN’s activation functions. Moreover, we opted for a multiple shots approach. So, it is clear that the EMPC optimization stage is nonlinear, hence generically non-convex. It possibly leads to a only local optimum [26]. Eq. (6c) is the plant state measure, Eq. (6d) is the state constraints and Eq. (6e) is the input constraints. Eq. (6f) is the contractive constraint, $P$ the contractive weighting matrix, and $\tilde{x}$ is the state deviation from the optimal steady state. The EMPC is computed at each iteration, and the first sample of the optimal input computed is applied to the plant’s actuators until the next sampling period. In addition, the EMPC defines an implicit feedback law:

$$\mu_N(\bar{x}) = u^*_0 \tag{7}$$

Where $\mu_N(\bar{x})$ is the implicit feedback related to $\bar{x}$ and $u^*_0$ is the optimal input computed.

### *3.4.2. Computing machine implementation*

The EMPC is translated and implemented in a modeling language specific to the field of optimization, and the optimization problem must be solved by an ad hoc solver. Several tools are available, such as JuMP [16] or Casadi [7]. The neural network used to define the state evolution constraint is a non-linear model, and thus a non-linear programming solver is needed, such as the Ipopt solver, which implements the interior point method [63]. Implementing such an on-line implicit feedback law $\mu_N(\bar{x})$ with high sampling can be computationally difficult [2], especially when the model is non-linear [14]. The optimization

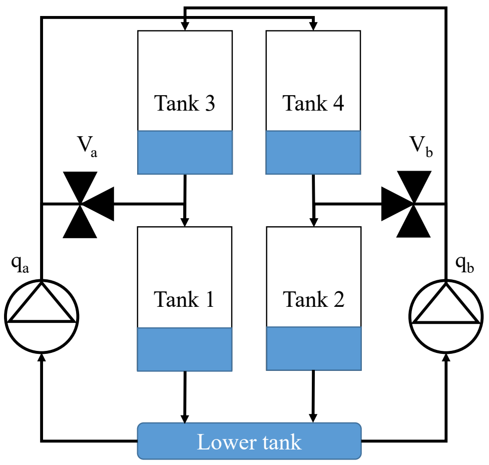

Figure 4: Four tank benchmark schematic view.

computation cannot exceed a sampling period. For some problems, a parallel implementation within computing machines composed of multi-core Central Processing Unit (CPU), GPU or Field-Programmable Gate Array (FPGA) [2] is necessary.

## 4. Example

### *4.1. Four tank benchmark*

To illustrate the relevance of the methodology, we apply it to a benchmark widely used for ordering: the level control of a Quadruple Tank Process (QTP), also known as the four-tank benchmark [34], is widely used by academics [33] and recognized for its capacity to illustrate industrial issues [58]. Contributions include the use of a decentralized PID controller [31] or a control by sliding mode in [53]. The implementation of a predictive control applied to the QTP benchmark is also widely discussed, and several distributed MPC methods are compared in [6]. The QTP benchmark has four tanks, two three-way valves and two pumps. The aim is to maintain a certain water level in the reservoirs. Pump $q_a$ fills tanks 1 and 4, then pump $q_b$ fills tanks 2 and 3. In addition, water leaks through tanks orifices, as tank 3 leaks to tank 1 and tank 4 to tank 2, then tanks 1 and 2 to lower tank, Fig. 4. As a result, the pump flows must be greater than zero. The main parameters of the benchmark are shown in Tab. 1.

### *4.2. Application of the proposed methodology*

#### *4.2.1. Physical modelling*

The QTP is firstly modelled using Modelica to support the simulation tests, which is fine modeling that considers water property, pumps inertia, pipe roughness and valve pressure loss. Each sub-model is graphically assembled graphically, see Fig. 5.

Table 1: QTP parameters [6].

| Parameters | Value | Units | Description |
|---|---|---|---|
| $h_{1_{max}}$ | 1.36 | m | Maximum water level tank 1 |
| $h_{2_{max}}$ | 1.36 | m | Maximum water level tank 2 |
| $h_{3_{max}}$ | 1.30 | m | Maximum water level tank 3 |
| $h_{4_{max}}$ | 1.30 | m | Maximum water level tank 4 |
| $h_{min}$ | 0.2 | m | Minimum water level all tanks |
| $q_{a_{max}}$ | 3.26 | $m^3/h$ | Maximum water flow pump a |
| $q_{b_{max}}$ | 4.00 | $m^3/h$ | Maximum water flow pump b |
| $q_{min}$ | 0 | $m^3/h$ | Minimum water flow all pumps |
| $S_c$ | 0.06 | m | All tanks cross-section |
| $a_1$ | $1.31 \times 10^{-4}$ | $m^2$ | Surface of the tank 1 leakage orifice |
| $a_2$ | $1.51 \times 10^{-4}$ | $m^2$ | Surface of the tank 2 leakage orifice |
| $a_3$ | $9.27 \times 10^{-5}$ | $m^2$ | Surface of the tank 3 leakage orifice |
| $a_4$ | $8.82 \times 10^{-5}$ | $m^2$ | Surface of the tank 4 leakage orifice |
| $V_a$ | 0.3 | - | Three way valve a opening |
| $V_b$ | 0.4 | - | Three way valve b opening |

Table 2: Parameters of the PRBS signals.

| Parameters | Value | Units |
|---|---|---|
| Time period | 500 | s |
| Start time | 500 | s |
| End time | 30 | days |
| Amplitude | $q_{min}$ or $q_{max}$ | $m^3/h$ |

### *4.2.2. Data generation*

The actuators of the dynamical system are the two pumps, and the measures are the four tanks water levels. Each water level in the reservoir can be viewed as a state variable: $\bar{x_i} = h_i$. The inputs are the pump water flows $\bar{u_i} = q_i$. Two random sequences were applied to the control inputs: The first is a PRBS signal, whose properties are provided in Tab. 2; the second is the piecewise constant with constant random values ranging from minimum to maximum water flow, see Tab. 3. The simulations were performed to generate a total of 10,368,000 samples for each neural network input with a 5s sample period. Furthermore, the data are filtered to only use data within water level constraints, which leads to 5,686,738 data samples available to feed the neural network.

### *4.2.3. Dynamical system identification*

The neural network inputs are water flows from pumps, and current water levels with back horizon $l = 0$, $v_k = \text{hcat}(q_{a_k}, q_{b_k}, h_{1_k}, h_{2_k}, h_{3_k}, h_{4_k})$. The out-

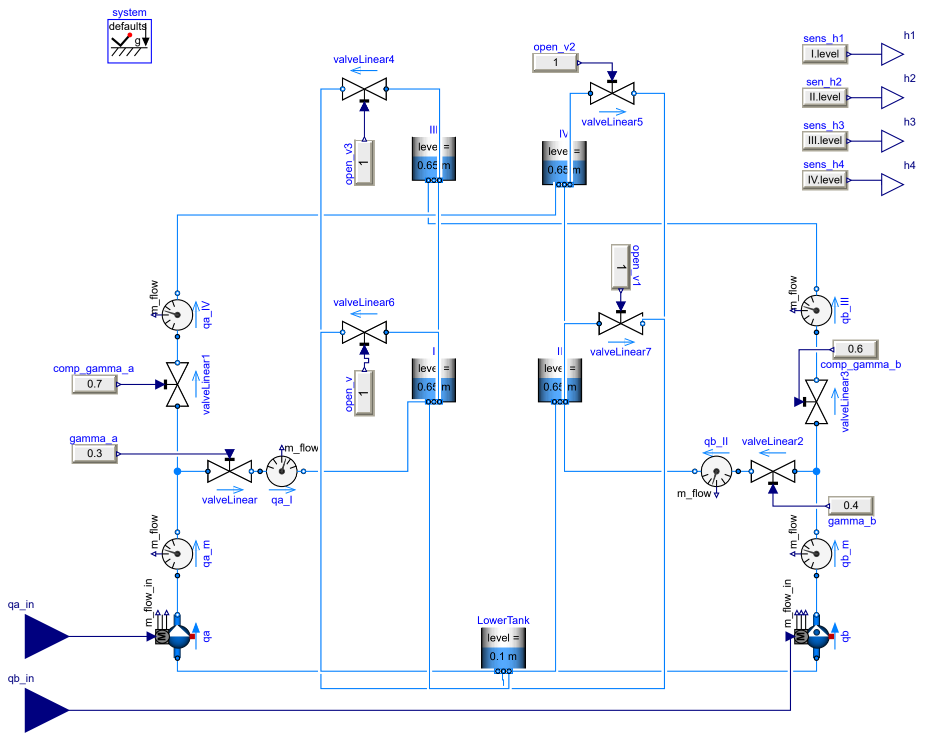

Figure 5: Graphical representation of the four-tank benchmark within Modelica in Dymola Software.

put of the neural network is the water levels predicted for the next sample time, $x_{k+1} = \text{hcat}(h_{1_{k+1}}, h_{2_{k+1}}, h_{3_{k+1}}, h_{4_{k+1}})$. The data generated are separated into two arrays for the training stage, where the first array contains the neural network inputs while the second has the outputs. The available data (5,686,738) are separated and randomized into 80% for training and 20% for testing, and during each of the 20 epochs, 10% of the training data are extracted for validation while the remainder feeds the neural network. Furthermore, the chosen loss function chosen is the MSE, and the neural networks were implemented with Flux.jl [32]. The meta-heuristic algorithm chosen to optimize the hyperparameters is a differential evolution algorithm [12] with adaptive setting [10]

Table 3: Parameters of constant input signals.

| Parameters | Value | Units |
|---|---|---|
| Time period | 1200 | s |
| Start time | 30 | days |
| End time | 600 | days |
| Amplitude | $q_{min}$ to $q_{max}$ | $m^3/h$ |

Table 4: Hyperparameters range values.

| Symbol | Description | Range value |
|---|---|---|
| $\sigma$ | Activation function | $\sigma \in$ [Bent [59], Gelu [28], Softplus [23], Swish [54]] |
| $\delta$ | Neurons number | $\delta \in$ [4 to 19] |
| $\theta$ | Hidden layer number | $\theta \in$ [1,2] |
| $\mu$ | Epoch number | $\mu \in$ [20 to 180] |
| $\nu$ | Batch size | $\nu \in$ [2048] |
| $o$ | Optimizer | $o \in$ [adam [38], radam [44], nadam [15]] |
| $\tau$ | Learning rate | $\tau \in [1 \times 10^{-7}$ to 0.01] |
| $\upsilon$ | Momentum exponential decay | $\upsilon \in$ [0.7 to 0.999] |
| $\phi$ | Momentum estimate | $\phi \in$ [0.7 to 0.999] |

and radius limited from Julia package BlackBoxOptim.jl [18].Hyperparameters range values are shown in Tab. 4. Otherwise, the neural network training were performed on Nvidia P6000 GPUs thanks to JuliaGPU [8].

### *4.2.4. EMPC tuning and implementation*

EMPC is optimized every 5s and over a time horizon equal to 25s (5 periods). The economic cost considered is a function of the squared weighted sum of the pump flow rates and the inverse of the water volumes in reservoirs one and two [35]. This reflects the objective of the search for two conflicting objectives: the maximization of the volume of water and the minimization of the energy consumption of the pumps (linked to the square of the flow rates):

$$l_e(x_k, u_k) = u_{1_k}^2 + \kappa_1 u_{2_k}^2 + \kappa_2 \frac{V_{min}}{S_c(x_{1_k} + x_{2_k})} \tag{8}$$

Where $\kappa_1$ and $\kappa_2$ are time varying costs. $V_{min} = 0.012m^3$ is the minimum volume of water in a tank, and $S_c$ is the tank cross-section. $\kappa_i = (\kappa_1; \kappa_2)$ values are equal to $\kappa_i = (1; 500)$ from 1 to 1,500s, $\kappa_i = (1; 1000)$ from 1,500 to 3,000s, $\kappa_i = (5; 100)$ from 3,000 to 4,500s and $\kappa_i = (0.5; 500)$ from 4,500 to 6,000s, Tab. 5. Also, the $P$ matrix selected is the identity matrix and the optimal steady state are visible in Tab. 5. Furthermore, constraints enable considering the water tanks' max level (state constraints) and admissible water flows (input constraints), Tab. 1. The predictive control is implemented within JuMP.jl with multiple shooting numerical method and the solver used is Ipopt within a workstation equipped with an Intel Xeon Gold 5122 CPU.

## *4.3. Simulation test bench*

The QTP fine-scale modelling from Modelica was exported using a Functional Mock-up Interface (FMI). A simulation software (Simulink) simulated the Functional Mock-up Unit (FMU) and saved states and inputs in text files throughout simulation. Furthermore, every 5s, the simulation was paused to

Table 5: $\kappa$ and optimal steady state values.

| Time | $\kappa_i$ | $x_1^*; x_2^*; x_3^*; x_4^*$ |
|---|---|---|
| 1 to 1,500 | (1; 500) | (0.519; 0.524; 0.514; 0.543) |
| 1,500 to 3,000 | (1; 1000) | (0.689; 0.675; 0.705; 0.668) |
| 3,000 to 4,500 | (5; 100) | (0.244; 0.291; 0.203; 0.370) |
| 4,500 to 6,000 | (0.5; 500) | (0.746; 0.512; 1.031; 0.260) |

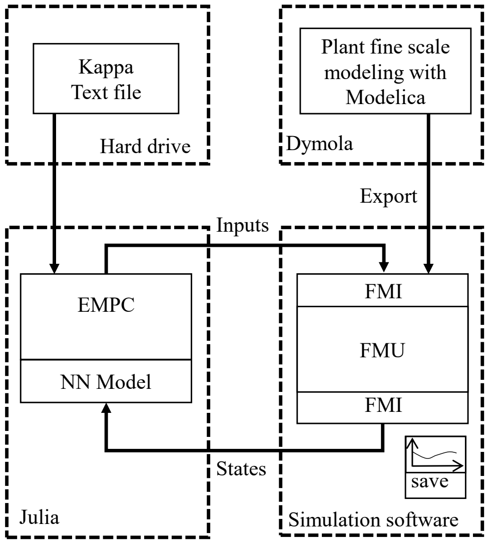

Figure 6: Controller and plant control in simulation.

execute the predictive control. Finally, the optimal control inputs computed were sent to the FMU and the simulation run again until the next iteration. A schematic representation is provided in Fig. 6.

## 5. Results

### *5.1. System identification*

Fig. 7.a depicts the boxplot hyperparameters optimization results with validation loss versus trained networks, where 618 neural networks were trained, 487 algorithm steps were performed and vanishing gradients occurred with 18 trained networks. The lowest validation loss is equal to $8.11 \times 10^{-7}$ and the highest is equal to $3.79 \times 10^{-1}$. The validation loss median is equal to $5.17 \times 10^{-6}$, and as a result, 50% of the trained networks have validation loss higher than $5.17 \times 10^{-6}$. Hyperparameters optimization allows selecting the best set of hyperparameters for a network with accurate identification. Fig. 7.b depicts the validation loss with clusters of 10 trained, which shows three groups: The first is from the first trained network until 200 trained networks. The boxplot validation loss slightly decreased and reached a lower validation loss at 200 trained

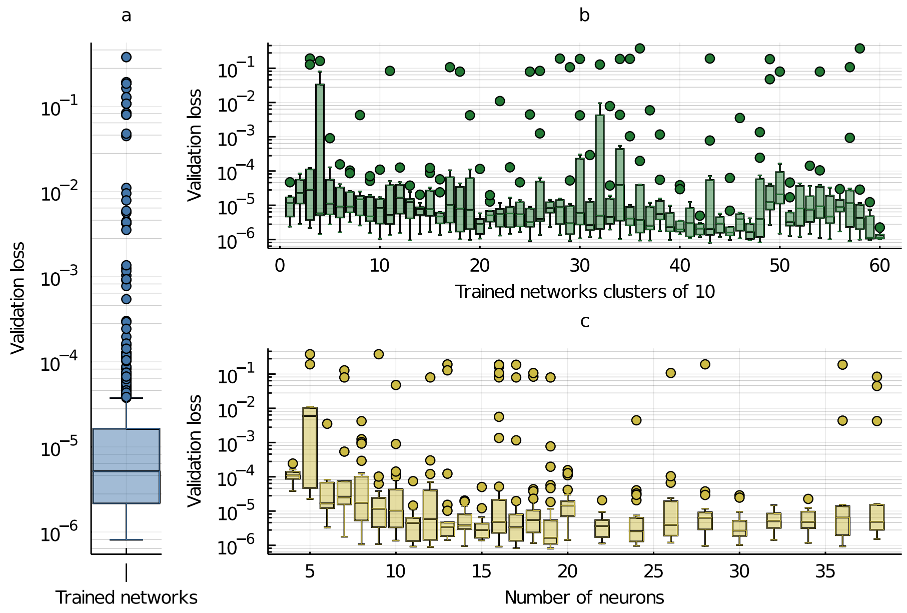

Figure 7: Cost function dependency on (a) the trained networks, (b) the trained networks with clusters of 10, (c) the neuron number.

networks. The second group, from 210 trained networks to 470 trained networks, validation loss increased and then decreased and reached lower validation loss at 470 trained networks. The third group, from 480 trained networks until the last trained networks, validation loss increased and then decreased and reached lower validation loss at 600 trained networks. This behavior shows that the meta-heuristic algorithm can select hyperparameters to reduce validation loss, but it needed a couple iterations. In addition, it reached some local minimal and tried to escape to find a lower validation loss. Furthermore, due to meta-heuristic optimization, it is not possible to answer whether the global minimal was reached. Fig. 7.c depicts validation loss with the total number of neurons considered within networks, which shows a validation loss reduction for network forms of 4 to 11 neurons; however, the validation loss plateaued for networks with more than 11 neurons. As a result, increasing beyond 11 neurons does not produce a reduction in validation loss but increases computational workload.

Fig. 8.a presents validation loss with activation functions. The medians are equal to $3.78 \times 10^{-6}$ for bent, $6.40 \times 10^{-6}$ for gelu, $8.26 \times 10^{-6}$ for softplus and $3.84 \times 10^{-6}$ for swish. Using bent or swish as activation functions allows a slight validation loss reduction compared to gelu or softplus. Fig. 8.b depicts the validation loss with hidden layers. The medians are equal to $4.34 \times 10^{-6}$ for one hidden layer and $6.37 \times 10^{-6}$ for two hidden layers. In this case, networks with one hidden layer allow reducing the validation loss, which is advantageous since neural networks with one hidden layer allow reducing network complexity

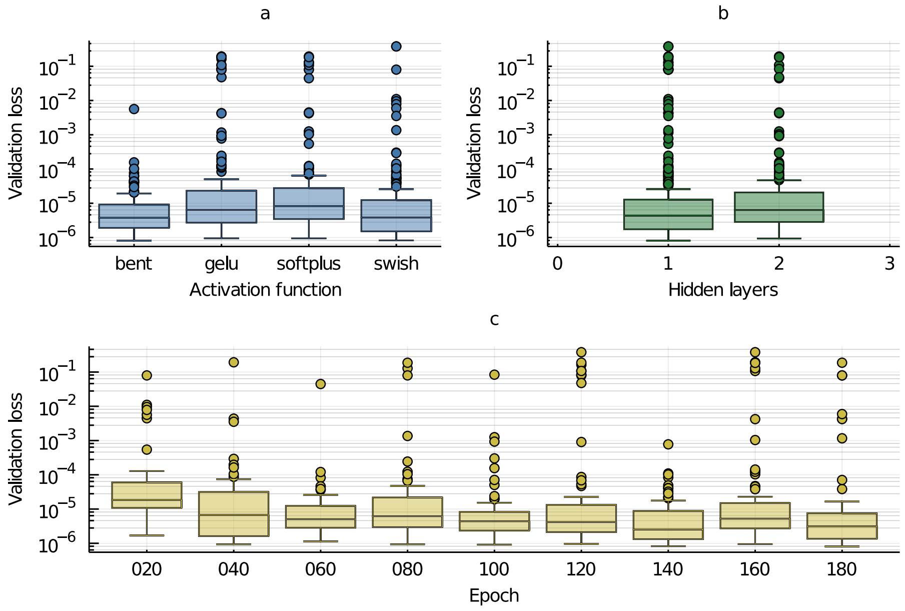

Figure 8: Cost function dependency on (a) the activation function considered, (b) the hidden layers number, (c) the epoch number.

and computational workload compared to networks with two hidden layers. Fig. 8.c depicts the validation loss and epochs and shows that increasing epoch beyond 40 is not necessary since the trained neural network median is equal to $1.84 \times 10^{-5}$ for 20 epochs and plateaued from 40 to 180 epochs between $5.00 \times 10^{-6}$ to $6.00 \times 10^{-6}$.

Fig. 9.a presents the validation loss with optimizers adam, nadam and radam. The validation loss medians are equal to $5.21 \times 10^{-6}$ for adam, $7.37 \times 10^{-6}$ for nadam and $3.50 \times 10^{-6}$ for radam. The radam reduced median validation loss compared to adam and nadam; however, because nadam and radam are derivative from the adam optimizer, the median validation loss remains similar, and no breakthroughs are noticeable. In addition, Fig. 9.b shows that the learning rate lower than $2.50 \times 10^{-3}$ allows reducing the median validation loss compared to learning rates higher than $2.50 \times 10^{-3}$. Furthermore, Fig. 9.c and Fig. 9.d show that momentums exceeding 0.9 allow reducing the validation loss. The median is equal to $2.49 \times 10^{-6}$ for the momentum exponential decay compared to $1.53 \times 10^{-5}$ and $6.71 \times 10^{-6}$. Additionally, median is equal to $3.01 \times 10^{-6}$ for momentum estimate compared to $1.35 \times 10^{-5}$ and $5.07 \times 10^{-6}$.

Table 6 presents the top 10 with trained neural networks which have the lowest validation loss, which shows that both one and two hidden layers are represented; however, there is only one network with two hidden layers. The number of neurons goes from 11 to 36, where increasing the number of neurons is not necessary, as shown in Fig. 7.c. The fourth activation functions are present

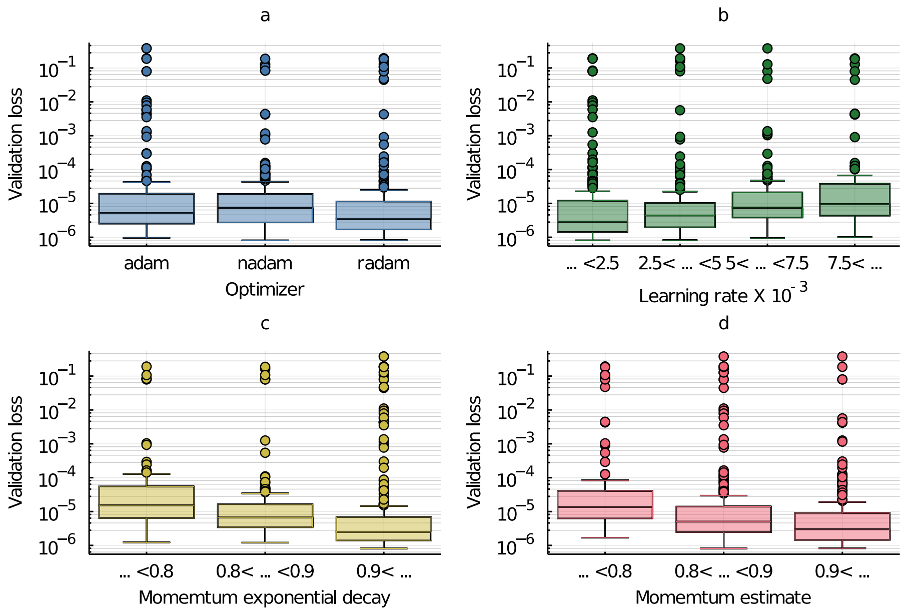

Figure 9: Cost function dependency on (a) the optimizer considered, (b) the learning rate, (c) the momentum exponential decay, (d) the momentum estimate.

within the top 10; however, there is a majority of swish with 6 over 10. In addition, the epoch number from the top 10 confirms the information extracted from Fig. 8.c, showing that increasing the number of the epoch is not necessary since there are trained networks with 40 or 80 epochs. Furthermore, radam and nadam prevail as optimizers, and adam is not represented within the top 10. Regarding learning rate, the results from Fig. 9.b are confirmed since a majority of the top 10 were trained with learning rates lower than $2.50 \times 10^{-3}$. Concerning momentums, the results from Fig. 9.c and Fig. 9.d are confirmed since most networks were trained with momentums higher than 0.9. Furthermore, the top 10 trained network validation loss goes from $9.46 \times 10^{-7}$ to $8.11 \times 10^{-7}$, which is close. Finally, we selected the top network for the EMPC (highlighted in bold in Table 6), and we calculated the trained loss and test loss. The results show a training loss equal to $8.22 \times 10^{-7}$ and test loss equal to $8.38 \times 10^{-7}$, and the network is not overfitted as test loss slightly increased compared to validation loss.

### *5.2. Process control*

Fig. 10.a depicts the simulation results with $h_1$ and $h_2$ water level controlled by the EMPC. Water levels $h_1$ and $h_2$ are controlled in an economic manner according to the economic cost given in Eq. (8), without using any explicit reference trajectory. The results show that the tanks' water levels stay within the constraints. In addition, $h_1$ and $h_2$ remain close when $\kappa_i = (1; 500)$ and

Table 6: Hyperparameters optimization results top 10 validation loss.

| - | Hidden layers | Amount of neurons | Activation function | Epoch | Optimizer | Learning rate | Momentum exponential decay | Momentum estimate | Validation loss |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **1** | **19** | **bent** | **180** | **nadam** | $\mathbf{2.18 \times 10^{-3}}$ | $\mathbf{9.97 \times 10^{-1}}$ | $\mathbf{8.97 \times 10^{-1}}$ | $\mathbf{8.11 \times 10^{-7}}$ |
| 2 | 1 | 17 | swish | 140 | radam | $2.71 \times 10^{-3}$ | $9.79 \times 10^{-1}$ | $9.22 \times 10^{-1}$ | $8.22 \times 10^{-7}$ |
| 3 | 1 | 12 | bent | 180 | nadam | $1.12 \times 10^{-4}$ | $9.82 \times 10^{-1}$ | $8.52 \times 10^{-1}$ | $8.93 \times 10^{-7}$ |
| 4 | 1 | 16 | swish | 100 | radam | $3.08 \times 10^{-3}$ | $9.79 \times 10^{-1}$ | $9.22 \times 10^{-1}$ | $9.14 \times 10^{-7}$ |
| 5 | 1 | 11 | swish | 140 | nadam | $2.71 \times 10^{-3}$ | $9.79 \times 10^{-1}$ | $9.22 \times 10^{-1}$ | $9.18 \times 10^{-7}$ |
| 6 | 1 | 11 | swish | 140 | nadam | $2.71 \times 10^{-3}$ | $9.79 \times 10^{-1}$ | $9.22 \times 10^{-1}$ | $9.19 \times 10^{-7}$ |
| 7 | 2 | 36 | swish | 40 | radam | $4.55 \times 10^{-4}$ | $9.72 \times 10^{-1}$ | $9.09 \times 10^{-1}$ | $9.35 \times 10^{-7}$ |
| 8 | 1 | 17 | swish | 80 | radam | $3.74 \times 10^{-3}$ | $9.84 \times 10^{-1}$ | $9.72 \times 10^{-1}$ | $9.35 \times 10^{-7}$ |
| 9 | 1 | 14 | softplus | 160 | radam | $5.58 \times 10^{-3}$ | $9.73 \times 10^{-1}$ | $9.82 \times 10^{-1}$ | $9.43 \times 10^{-7}$ |
| 10 | 1 | 16 | gelu | 180 | radam | $3.16 \times 10^{-4}$ | $9.07 \times 10^{-1}$ | $9.32 \times 10^{-1}$ | $9.46 \times 10^{-7}$ |

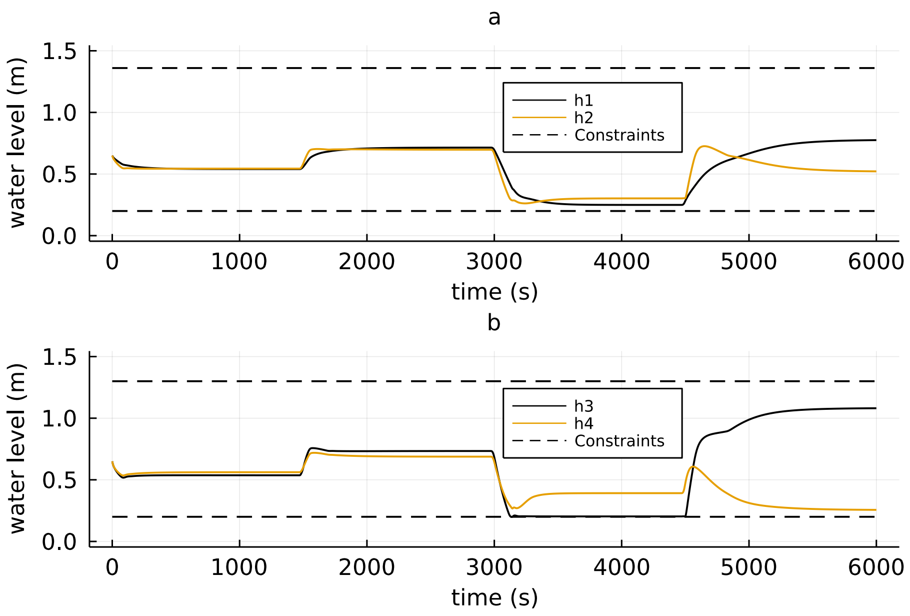

Figure 10: Simulation results. (a) Depicts tank 1 and 2 water levels over time. (b) Depicts tank 3 and 4 water levels over time.

$\kappa_i = (1; 1000)$. Furthermore, when the cost function weights change from $\kappa_i = (1; 500)$ to $\kappa_i = (1; 1000)$ at 1,500s, the water levels $h_1$ and $h_2$ are both increased, which is the expected behavior. Moreover, $h_1$ and $h_2$ pull apart as expected when using distinct weights associated with the different levels $\kappa_i = (5; 100)$ or $\kappa_i = (0.5; 500)$. Fig. 10.b depicts the simulation results with $h_3$ and $h_4$ water level controlled by EMPC. Once again, water levels $h_3$ and $h_4$ are controlled economically within the economic cost function defined by Eq. (8). The latitude of permissible water levels for these reservoirs is exploited while remaining within the established limits. It can be observed that $h_3$ is equal to the minimum water

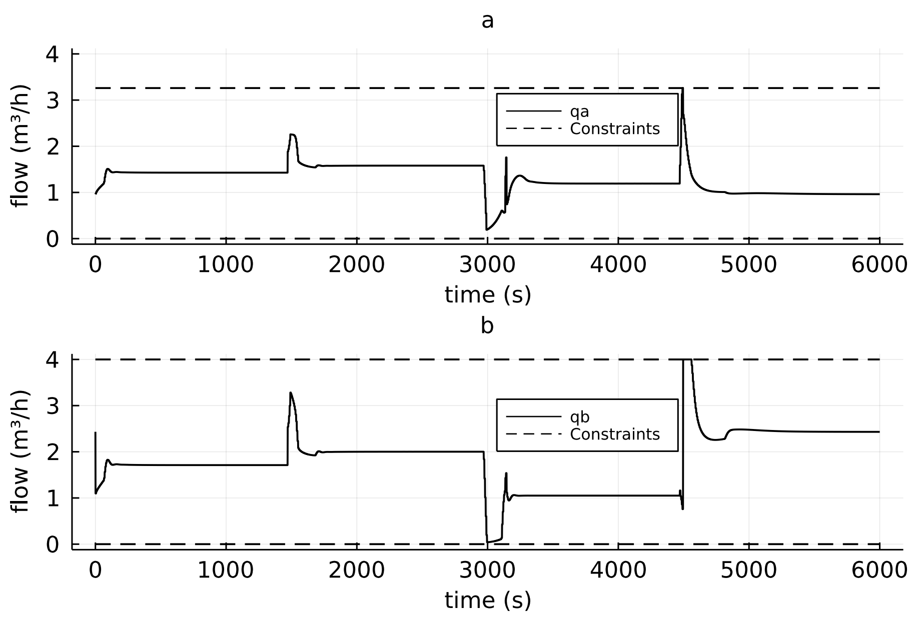

Figure 11: Simulation results. (a) Depicts pump a flow control over time. (b) Depicts pump b flow control over time.

level constraint from 3,000 to 4,500s. In addition, $h_3$ and $h_4$ remain close when $\kappa_i = (1; 500)$ and $\kappa_i = (1; 1000)$ and $h_3$ and $h_4$ pull apart when $\kappa_i = (5; 100)$ and $\kappa_i = (0.5; 500)$.

Fig. 11.a depicts the $q_a$ pump flow controlled by the EMPC. During the simulation, the pump flow stays within constraints. When the cost index $\kappa_i$ changes from (1;500) to (1;1000), the pump flow increases. In addition, when the cost index $\kappa_i$ change from (1;1000) to (5;100), the pump flow decreases. However, at 3,250s, it can be seen a peak on the pump flow $q_a$. This behavior is related to the $h_3$ state's constraint, during which time the pump flow increases to avoid violating the water level constraint. Finally, when the cost index $\kappa_i$ changes from (5;100) to (0.5;500), pump $q_a$ flow increases to maximum flow and then decreases and plateaus until the end of the simulation. Fig. 11.b depicts the $q_b$ pump flow controlled by the EMPC. During all simulations, the pump flow stays within constraints. When the cost index $\kappa_i$ changes from (1;500) to (1;1000), the pump flow increases. In addition, when the cost index $\kappa_i$ changes from (1,1000) to (5,100), the pump flow decreases to 0 $m^3/h$. At 3250s however, it can be seen a peak on the pump flow $q_b$. This behavior, identical to the previous case, is due to the state constraint $h_3$, during which time the pump flow increases to avoid breaking the water level constraint. Finally, when the cost index $\kappa_i$ changes from (5;100) to (0.5;500), pump $q_b$ flow increases to maximum flow and then decreases and plateaus at 2.4 $m^3/h$ until the end of the simulation, compared to $q_a$ which plateaus at 1 $m^3/h$.

## 6. Conclusion

This study aimed to propose an integrated methodology for non-linear predictive control design, synthesized from a neural network based model identified thanks to simulated data from multi-physics modelling. Results showed that the Quadruple Tank Process benchmark considered was well identified thanks to a neural network with optimized hyperparameters. Simulation using the predictive controller designed thanks to the methodology proposed depicts the expected behavior, in total consistency with the economic cost used. The presented methodology features interesting characteristics, such as being based on a fundamental movement in modeling, which consists of using bricks from existing physical models to constitute the model of the process considered. This supposes that such bricks exist, but this assumption becomes more reasonable every day, since a large community enriches the existing library.Based on this observation, the goal is to reduce the experiments providing data likely to calibrate the model, which thus performs appropriate simulations to produce the ad hoc data. On this basis, the model identifies an FNN which approaches the behavior of the physical process as closely as possible, while being less expensive to simulate. Choosing the characteristics of the target neural network (architecture and components among others) is effectively achieved by proceeding to the automated choice of the hyperparameters of the neural network by means of an adapted heuristic. Finally, the desired predictive control is obtained by choosing an economic criterion (EMPC), which is efficiently optimized thanks to choosing an ad hoc optimizer and using the FNN as an internal model. This method has been successfully implemented to control the QTP benchmark, for which different MPC solutions have already been tested in the control literature.

Among the perspectives, we are currently working to demonstrate the interest of this methodology applied to the generic problem of controlling multi-energy network systems [47]. One perspective in order to avoid possible risks for the MPC control, i) to be satisfied with local minima and ii) to accelerate the resolution consists, in reformulating the optimization problem in convex and conic constraints with a disciplined convex programming [61]; in this case, restrictions must be added to the neural network [9].

## Acknowledgment

This research was funded by Conseil Régional des Pays de la Loire, France. The authors thank the Julia community for online comments about the Julia language and packages.

## Data availability statement

Research data are not shared.

## References

[1] M. Abdel-Basset, L. Abdel-Fatah and A. K. Sangaiah, “ Metaheuristic Algorithms: A Comprehensive Review,” in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, A. K. Sangaiah, M. Sheng and Z. Zhang, Eds., Academic Press, 2018, ch. 10, pp. 185–231, doi: 10.1016/B978-0-12-813314-9.00010-4.

[2] K. M. Abughalieh and S. G. Alawneh, “A Survey of Parallel Implementations for Model Predictive Control,” *IEEE Access*, vol. 7, pp. 34348–34360, 2019, doi: 10.1109/ACCESS.2019.2904240.

[3] A. Afram and F. Janabi-Sharifi, “Theory and applications of HVAC control systems – A review of model predictive control (MPC),” *Buil. Environ.*, vol. 72, pp. 343–355, Feb. 2014, doi: 10.1016/j.buildenv.2013.11.016.

[4] K.S. Agbli *et al.*, “Multiphysics simulation of a PEM electrolyser: Energetic Macroscopic Representation approach,” *Int. J. Hydrogen Energy*, vol. 36, no. 2, pp. 1382–1398, Jan. 2011, doi: 10.1016/j.ijhydene.2010.10.069.

[5] Q. Altes-Buch, S. Quoilin and V. Lemort, “Greenhouses: a modelica library for the simulation of greenhouse climate and energy systems,” in *Proc. 13th International Modelica Conference*, Regensburg, Germany, 4-6 Mar. 2019, pp. 533–542, doi: 10.3384/ecp19157533.

[6] I. Alvarado *et al.*, “A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark,” *J. Process Control*, vol. 21, no. 5, pp. 800–815, Jun. 2011, doi: 10.1016/j.jprocont.2011.03.003.

[7] J. A. Andersson *et al.*, “CasADi: a software framework for nonlinear optimization and optimal control,” *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019, doi: 10.1007/s12532-018-0139-4.

[8] T. Besard, C. Foket, and B. De Sutter, “Effective extensible programming: unleashing Julia on GPUs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 827–841, Apr. 2019, doi: 10.1109/TPDS.2018.2872064.

[9] Y. Chen, Y. Shi, and B. Zhang, “Optimal Control Via Neural Networks: A Convex Approach,” *Proc. 7th International Conference on Learning Representations*, New Orleans, LA, USA, 6-9 May 2019.

[10] R. D. Al-Dabbagh *et al.*, “Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy,” *Swarm Evol. Comput.*, vol. 43, pp. 284–311, Dec. 2018, doi: 10.1016/j.swevo.2018.03.008.

[11] M. L. Darby and M. Nikolaou, “MPC: current practice and challenges,” *Control Eng. Pract.*, vol. 20, no. 4, pp. 328–342, Apr. 2012, doi: 10.1016/j.conengprac.2011.12.004.

[12] S. Das, S. S. Mullick and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016, doi: 10.1016/j.swevo.2016.01.004.

[13] S. L. de Oliveira Kothare and M. Morari, "Contractive model predictive control for constrained nonlinear systems, " *IEEE Trans. Autom. Control.*, vol. 45, no. 6, pp. 1053–1071, Jun. 2000, doi: 10.1109/9.863592.

[14] M. Diehl, H. J. Ferreau and N. Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, L. Magni, D. M. Raimondo, and F. Allgöwer, Éds., Berlin, Springer, 2009, pp. 391–417.

[15] T. Dozat, "Incorporating nesterov momentum into adam," in *Workshop track - International Conference on Learning Representations*, San Juan, Puerto Rico, 2-4 May 2016.

[16] I. Dumming, J. Huchette and M. Lubin, "JuMP: a modeling language for mathematical optimization," *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, May 2017, doi: 10.1137/15M1020575.

[17] M. Ellis, H. Durand and P. D. Christofides, "A tutorial review of economic model predictive control methods," *J. Process Control*, vol. 24, no. 8, pp. 1156–1178, Aug. 2014, doi: 10.1016/j.jprocont.2014.03.010.

[18] R. Feldt and A. Stukalov, "BlackBoxOptim.jl," *GitHub repository*, 2018, [Online], available: https://github.com/robertfeldt/BlackBoxOptim.jl.

[19] F. A. Fortin *et al.*, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp 2171–2175, 2012.

[20] R. Franke and H. Wiesmann, "Flexible modeling of electrical power systems–the Modelica PowerSystems library," in *Proc. 10th International Modelica Conference*, Lund, Sweden, 10-12 Mar. 2014, pp. 515–522, doi: 10.3384/ecp14096515.

[21] P. J. Gawthrop and G. P. Bevan, "Bond-graph modeling," *IEEE Control Sys. Mag.*, vol. 27, no. 2, pp. 24–45, Apr. 2007, doi: 10.1109/MCS.2007.338279.

[22] L. Giraud *et al*, "Presentation, validation and application of the District Heating modelica library," in *Proc. 11th International Modelica Conference*, Versailles, France, 21-23 Sep., 2015, pp. 79–88, doi: 10.3384/ecp1511879

[23] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. $14^th$ International Conference Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, vol. 15, pp. 315–323, Apr. 2011.

[24] L. Grüne, "Economic receding horizon control without terminal constraints," *Automatica*, vol. 49, no. 3, pp. 725–734, Mar. 2013, doi: 10.1016/j.automatica.2012.12.003.

[25] L. Grüne and J. Pannel, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control: Theory and Algorithms*, Springer, Cham, 2017, ch. 3, pp. 45–69, doi: 10.1007/978-3-319-46024-6_3.

[26] L. Grüne and J. Pannek, "Numerical Optimal Control of Nonlinear Systems," in *Nonlinear Model Predictive Control: Theory and Algorithms*, Springer, Cham, 2017, ch. 12, pp. 367–434, doi: 10.1007/978-3-319-46024-6_12.

[27] B. El Hefni, D. Bouskela and G. Lebreton, "Dynamic modelling of a combined cycle power plant with ThermoSysPro," in *Proc. 8th International Modelica Conference*, Dresden; Germany, 20-22 Mar., 2011, pp. 365–375, doi: 10.3384/ecp11063365.

[28] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, accessed: 2020, [Online], available: https://arxiv.org/abs/1606.08415.

[29] D. Hrovat *et al.*, "The development of model predictive control in automotive industry: a survey," in *Proc. IEEE Int. Conf. Control Appl.*, Dubrovnik, Croatia, Oct. 3–5, 2012, pp. 295–302, doi: 10.1109/CCA.2012.6402735.

[30] J. K. Huusom *et al.*, "Tuning of methods for offset free MPC based on ARX model representations," in *Proc.s of the 2010 Am. Control Conf.*, Baltimore, MD, USA, 30 Jun.-2 Jul., 2010, pp. 2355–2360, doi: 10.1109/ACC.2010.5530560.

[31] P. Hušek, "Decentralized PI Controller Design Based on Phase Margin Specifications,"" *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 1, pp. 346–351, Jan. 2014, doi: 10.1109/TCST.2013.2248060.

[32] M. Innes, "Flux: elegant machine learning with Julia," *J. Open Source Softw.*, vol. 3, no. 25, art no. 602, May 2018, doi: 10.21105/joss.00602.

[33] K. H. Johansson *et al.*, "Teaching multivariable control using the quadruple-tank process," in *Proc. 38th IEEE Conf. Decis. Control*, Vol. 1, pp. 807–812, Dec. 1999, doi: 10.1109/CDC.1999.832889.

[34] K. H. Johansson, "The quadruple-tank process: a multivariable laboratory process with an adjustable zero," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 456–465, May 2000, doi: 10.1109/87.845876.

[35] A. D' Jorge *et al*, "A robust economic MPC for changing economic criterion," *Int. J. Robust Nonlinear Control*, vol. 28, no. 15, pp. 4404–4423, Oct. 2018, doi: 10.1002/rnc.4243.

[36] J. B. Jørgensen *et al.*, "Economic MPC for a linear stochastic system of energy units," in *Proc. Control Conf. ECC Eur.*, Aalborg, Denmark, 29 Jun.- 1 Jul. 2016, pp. 903–909, doi: 10.1109/ECC.2016.7810404.

[37] K. K. K. Kim, E. R. Patrón and R. D. Braatz, “Standard representation and unified stability analysis for dynamic artificial neural network models,” *Neural Netw.*, vol. 98, pp. 251–262, Feb. 2018, doi: 10.1016/j.neunet.2017.11.014.

[38] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. 3rd International Conference on Learning Representations*, San Diego, CA, USA, 7-9 May 2015.

[39] I. D. Landau and G. Zito, “System identification: the bases,” in *Digital control systems. Design, identification and implementation*, Springer, London, 2006, doi: 10.1007/978-1-84628-056-6_5

[40] F. Lamnabhi-Lagarrigue *et al.*, “Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges,” *Annu. Rev. Control*, vol. 43, pp. 1–64, Apr. 2017, doi: 10.1016/j.arcontrol.2017.04.001.

[41] M. Lazar, “Model predictive control of hybrid systems : stability and robustness,” Ph.D. dissertation, Technische Universiteit Eindhoven, Eindhoven , 2006, doi: 10.6100/IR612103.

[42] A. Lefort *et al.*, “Hierarchical control method applied to energy management of a residential house,” *Energy Buil.*, vol. 64, pp. 53–61, Sept. 2013, doi: 10.1016/j.enbuild.2013.04.010.

[43] G. Li and J. Shi, “On comparing three artificial neural networks for wind speed forecasting,” *Appl. Energy*, vol. 87, no. 7, pp. 2313–2320, Jul. 2010, doi: 10.1016/j.apenergy.2009.12.013.

[44] L. Liu *et al.*, “On the Variance of the Adaptive Learning Rate and Beyond,” in *Proc. 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 26-30 Apr. 2020.

[45] L. Liu, F. Felgner and G. Frey, “Comparison of 4 numerical solvers for stiff and hybrid systems simulation,” in *Proc. IEEE 15th Conf. Emerg. Technol. & Factory Automat. (ETFA)*, Bilbao, Spain, 13-16 Sept., 2010, pp. 1–8, doi: 10.1109/ETFA.2010.5641330.

[46] E. T. Maddalena, Y. Lian and C. N. Jones, “Data-driven methods for building control — A review and promising future directions,” *Control Eng. Pract.*, vol. 95, art no. 104211, Feb. 2020, doi: 10.1016/j.conengprac.2019.104211.

[47] M. J. O’Malley *et al*, “Multicarrier energy systems: shaping our energy future”, *Proc. IEEE*, vol. 108, no. 9, pp. 1437–1456, Sept. 2020, doi: 10.1109/JPROC.2020.2992251.

[48] S. E. Mattsson, H. Elmqvist and M. Otter, “Physical system modeling with Modelica,” *Control Eng. Pract.*, vol. 6, no. 4, pp. 501–510, Apr. 1998, doi: 10.1016/S0967-0661(98)00047-1.

[49] D. Q. Mayne, “Model predictive control: recent developments and future promise,” *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014, doi: 10.1016/j.automatica.2014.10.128.

[50] M. Murillo, G Sánchez and L Giovanini, “Iterated non-linear model predictive control based on tubes and contractive constraints,” *ISA Trans.*, vol. 62, pp. 120–128, May 2016, doi: 10.1016/j.isatra.2016.01.008.

[51] K. R. Muske and J. B. Rawlings, “Model predictive control with linear models,” *AIChE J.*, vol. 39, no. 2, pp. 262–287, Feb. 1993, doi: 10.1002/aic.690390208.

[52] A. Paszke *et al.*, “Pytorch: an imperative style, high-performance deep learning library,” *Adv. Neural Inf. Process Syst.*, pp. 8026–8037, 2019.

[53] S. B. Prusty *et al.*, “Sliding mode control of coupled tank systems using conditional integrators,” *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 1, pp. 118–125, Jan. 2020, doi: 10.1109/JAS.2019.1911831.

[54] P. Ramachandran, B. Zoph and Q. V. Le, “Searching for activation functions,” in *Proc. $6^{th}$ International Conference on Learning Representations*, Vancouver, BC, Canada, 30 Apr. - 3 May, 2018.

[55] J. B. Rawlings, D. Angeli and C.N. Bates, “Fundamentals of economic model predictive control,” in *Proc. 51st IEEE Conf. Decision Control (CDC)*, Maui, HI, Dec. 10-13, 2012, pp. 3851–3861, doi: 10.1109/CDC.2012.6425822.

[56] J. Richalet *et al.*, “Model predictive heuristic control: application to industrial processes,” *Automatica*, vol. 14, no. 5, pp. 413–428, Sept. 1978, doi: 10.1016/0005-1098(78)90001-8.

[57] P. Schrangl, P. Tkachenko and L. del Re, “Iterative Model Identification of Nonlinear Systems of Unknown Structure: Systematic Data-Based Modeling Utilizing Design of Experiments,” *IEEE Control Sys. Mag.*, vol. 40, no. 3, pp. 26–48, Jun. 2020, doi: 10.1109/MCS.2020.2976388.

[58] D. Shneiderman and Z. J. Palmor, “Properties and control of the quadruple-tank process with multivariable dead-times,” *J. Process Control*, vol. 20, no. 1, pp. 18–28, 2009, doi: 10.1016/j.jprocont.2009.10.010.

[59] D. Smith and H. LeBlanc, “Variable Activation Functions and Spawning in Neuroevolution,” in *Proc. 2018 ASEE North Central Section Conference*, Akron, Ohio, USA, 2018.

[60] E. Terzi *et al.*, "Learning-based predictive control of the cooling system of a large business centre," *Control Eng. Pract.*, vol. 97, art no. 104348, Apr. 2020, doi: 10.1016/j.conengprac.2020.104348.

[61] M. Udell *et al.*, "Convex optimization in Julia", *First Workshop for High Performance Technical Computing in Dynamic Languages*, New Orleans, LA, USA, 17-17 Nov. 2014, doi: 10.1109/HPTCDL.2014.5.

[62] A. Vasilyev *et al.*, "Component-based modelling of PEM fuel cells with bond graphs," *Int. J. Hydrogen Energy*, vol. 42, no. 49, pp. 29406–29421, Dec. 2017, doi: 10.1016/j.ijhydene.2017.09.004.

[63] A. Wächter and T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006, doi: 10.1007/s10107-004-0559-y.

[64] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Proc. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009, doi: 10.1109/MSP.2008.930649.

[65] A. Weiss *et al.*, "Model predictive control for spacecraft rendezvous and docking: strategies for handling constraints and case studies," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1638–1647, Jul. 2015, doi: 10.1109/TCST.2014.2379639.

[66] M. Wetter *et al.*, "Modelica buildings library," *J. Build. Perform. Simul.*, vol. 7, no. 4, pp. 253–270, 2014, doi: 10.1080/19401493.2013.765506.

[67] W. Wong *et al.*, "Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing," *Mathematics*, vol. 6, no. 11, art no. 242, Nov. 2018, doi: 10.3390/math6110242.

[68] Z. Wu and P. D. Christofides, "Economic Machine-Learning-Based Predictive Control of Nonlinear Systems, " *Mathematics*, vol. 7, no. 6, art no. 494, Jun. 2019, doi: 10.3390/math7060494.

[69] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[70] X. S. Yang, Nature-inspired metaheuristic algorithms, Frome, United Kingdom, pp. 117–118, 2010.

[71] B.L. Ye *et al.*, "A Survey of Model Predictive Control Methods for Traffic Signal Control," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 623–640, May 2019, doi: 10.1109/JAS.2019.1911471.

[72] J. Zhang and I. Mitliagkas, "Yellowfin and the art of momentum tuning," *Proc. 2nd SysML Conference*, Palo Alto, CA, USA, 31 Mar. - 2 Apr., 2019.